INFORMATION PROCESSING APPARATUS, INFORMATION PROCESSING METHOD, AND COMPUTER-READABLE MEMORY MEDIUM STORING PROGRAM FOR REALIZING THE METHOD

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to peripheral devices, an information processing apparatus in which a control program for controlling the peripheral devices has been installed, and a method and a control program for controlling those devices and apparatus.

Related Background Art

A program group called an installation set is necessary for installing a device driver (hereinafter, referred to as a driver) as a control program for controlling peripheral devices. The driver as an installation target, an installer as a control program for installing the driver, and the like are included in the installation set. The drivers ordinarily differ in dependence on a type or a version. As an installation set, there are two types of formats: a format in which a plurality of corresponding model types are incorporated in one installation set; and a format in which the installation set is formed every corresponding model type. Specifically speaking, in the case of the former format, when installation start is instructed to an installing program called an installer, a plurality of model types are listed as installation targets and one of them can be selected. In the case of the latter format, only one model type is selected as an installation target upon installation and the driver can be installed in response to an instruction of the user. However, if the drivers of a plurality of corresponding model types are incorporated into one installation set in the former format, the following inconvenience is caused. For example, if a driver named BBB which corresponds to a version 2.00 is installed into a personal computer in which a driver named AAA whose version is 1.00 has been installed, a problem such that the version of the driver corresponding to AAA is also automatically upgraded to v2.00 occurs. The problem as mentioned above occurs particularly in the case where AAA and BBB include the same driver module group.

Although the above method is very advantageous to the user who always wants to upgrade the version of a printer driver installed in the computer to the latest, it means the version of the printer driver is arbitrarily upgraded without the user's recognition or consent.

However, the foregoing method is disadvantageous to the recent users who spend a long time to test the operation of the printer drivers to confirm it upon installation of a system and use only the drivers whose operation has been confirmed in a user environment. For example, in a large office which uses a print application or the like which depends largely on the printer driver in combination with the printer driver, severely strict management of the versions of the drivers is required and there is a strong demand to upgrade the version of only the driver designated by the user. There is also a strong demand to selectively and properly upgrade the version of the driver as a target of the version-up. This is true of management of the device drivers of the peripheral devices other than a printer.

For example, if the module sets constructing the printer drivers are the same in the above case, there is a case where the module of AAA is overwritten accompanied with the new installation of the new printer driver BBB. There is also a disadvantage such that the system has to be re-activated after the installation due to the overwriting process.

SUMMARY OF THE INVENTION

There is a strong demand to avoid the re-activation of the system in a computer of a server system in the large office as much as possible. The user who wants to avoid the re-activation accompanied with the unnecessary upgrade of the modules which is caused by the new installation of a certain module as mentioned above.

If the installation set is formed every corresponding model type, since the module sets constructing the printer drivers differ every model type, the system re-activation after the installation of the new printer driver and the version-up which is not desired by the user as mentioned above can be prevented. There is, however, a disadvantage such that since it is necessary to form the printer driver every model type, the number of developing steps is very large. Particularly, when the apparatuses cope with a new OS (Operating System), or the like, it is necessary to form the printer drivers of all model types, and when it is intended that contents of a fault corrected in a certain model type are reflected to another model type, it is necessary to separately form the printer driver of such another model type. If there are apparatuses of many model types whose version-up is desired, it is necessary to prepare the installation sets of the apparatuses of the model types which need the version-up and separately execute them, so that the installing operation is complicated.

The invention intends to solve such a situation and it is an object of the invention to provide a mechanism such that when a device driver is installed, names of module sets to be installed are changed in accordance with a predetermined rule, thereby properly installing the driver.

Another object of the invention is to provide a mechanism such that a driver as a target of version-up can be properly selected and the version of the selected driver can be properly upgraded.

Further another object of the invention is to provide a mechanism such that even in the case of newly installing a certain driver, another driver is not overwritten due to such installation, thereby reducing the number of times of re-activation.

According to the first aspect of the invention, there is provided an information processing apparatus which can install a first control program corresponding to a first peripheral device and a second control program for controlling a second peripheral device, the first and second control programs including a common module. The apparatus comprises deciding means for deciding so that identification information of the common module which operates as a part of the first control program and identification information of the common module which operates as a part of the second control program are made different.

Other features and advantages of the present invention will be apparent from the following description taken in conjunction with the accompanying drawings, in which like reference characters designate the same or similar parts throughout thereof.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram for explaining a construction of a print control apparatus showing an embodiment of the invention;

Fig. 2 is a block diagram for explaining a typical print data forming method in a host computer;

Fig. 3 is a block diagram for explaining a print data forming method which is used for forming an intermediate code and is a diagram obtained by expanding Fig. 2;

Fig. 4 shows a schematic diagram of a printer driver installing method for explaining a whole construction of the embodiment;

Fig. 5 shows a correlation diagram between friendly names of printer drivers and hexadecimal numbers which are formed by CRC.exe;

Fig. 6 shows an example of an INF file for a certain OS which is used in the embodiment;

Fig. 7 shows an example of an INF file for another OS which is used in the embodiment and different from Fig. 6;

Fig. 8 shows necessity of system re-activation after installation;

Fig. 9 is a diagram showing an example of rename of a common module;

Fig. 10 is a diagram showing a description example of an INF file which can be used in a certain OS;

Fig. 11 is a diagram showing a description example of an INF file which can be applied to another OS different from Fig. 10;

Fig. 12 is a diagram showing an example of rename of a common module;

Fig. 13 is a diagram showing an example of rename of a common module;

Fig. 14 is a diagram showing a description example of an INF file which can be used in a certain OS;

Fig. 15 is a diagram showing an example of an INF file which can be applied to another OS different from Fig. 14;

Fig. 16 is a diagram showing an example of rename of a common module;

Fig. 17 is a diagram showing a description example of an INF file which can be used in a certain OS;

Fig. 18 is a diagram showing an example of an INF file which can be applied to another OS different from Fig. 14; and

Fig. 19 is a block diagram showing an example of modules in a host computer 3000.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Embodiments of the invention will be described hereinbelow with reference to the drawings. The following embodiments are examples for explaining various aspects of the invention and, naturally, the invention is not limited to the following embodiments within the purview without departing from the spirit of the invention.

(First embodiment)

Fig. 1 is a diagram showing a construction of a printer control system showing an example of the embodiment of the invention. Naturally, the invention can be applied to any of a single apparatus, a system comprising a plurality of apparatuses, and a system which is connected via a network such as LAN, WAN, or the like and in which processes are executed so long as the functions of the invention are executed. In the diagram, a host computer 3000 has a CPU 1 for executing processes of a document in which a figure, an image, characters, a table (including a spreadsheet, etc.), and the like exist mixedly on the basis of a document processing program or the like stored in a program ROM of a ROM 3 or an external memory 11. The CPU 1 integratedly controls each device connected to a system bus 4. An operating system program (hereinafter, abbreviated to OS) or the like as a control program of the CPU 1 is stored in the program ROM of the ROM 3 or the external memory 11. Font data or the like which is used in the document processes is stored in a font ROM of the ROM 3 or the external memory 11. Various data which is used when the document processes or the like are executed is stored in a data ROM of the ROM 3 or the external memory 11. A RAM 2 functions as a main memory, a work area, or the like of the CPU 1. An installer as a control program to install the printer drivers is also stored in the HDD 11. The OS is controlled in a manner such that the installer is loaded into the RAM 2, the driver is read out from the HDD 11 and obtained or the driver is obtained via a network (not shown), and the printer driver obtained by the installer is stored into a predetermined storing area via an API of the OS. Further, the installer is registered into an area called a registry via the API of the OS so that the OS can recognize the driver which has newly been installed.

A keyboard controller (KBC) 5 controls a key input from a keyboard 9 or a pointing device (not shown). A CRT controller (CRTC) 6 controls a display of a CRT display (CRT) 10. A disk controller (DKC) 7 controls an access to the external memory 11 such as a hard disk (HD), a floppy (registered trademark) disk (FD), or the like for storing a boot program, various applications, font data, a user file, an edit file, a printer control command generating program (hereinafter, referred to as a printer driver), and the like. A printer controller (PRTC) 8 is connected to a printer 1500 via a bidirectional interface (interface) 21 and executes a communication control process with the printer 1500. Although the printer driver has been mentioned as a control program for controlling the peripheral devices in the embodiment, a control program for controlling a digital camera, a copying apparatus, a facsimile, an ink jet printer, a laser beam printer, or their hybrid apparatus can be also used.

The CPU 1 executes, for example, a developing (rasterizing) process of an outline font into a display information RAM set on the RAM 2, thereby enabling WYSIWYG on the CRT 10 to be realized. The CPU 1 opens various registered windows on the basis of commands instructed by a mouse cursor (not shown) or the like on the CRT 10 and executes various data processes. When the user executes the print, the CPU 1 opens the window regarding the print setting, thereby enabling the user to set a printer or set a print processing method for the printer driver including selection of a printing mode. The printer 1500 is controlled by a CPU 12. The printer CPU 12 outputs an image signal as output information to a printing unit (printer engine) 17 connected to a system bus 15 on the basis of a control program or the like stored in a program ROM of a ROM 13 or a control program or the like stored in an external memory 14. A control program or the like of the CPU 12 is stored in the program ROM of the ROM 13. Font data or the like which is used when the output information is formed is stored in a font ROM of the ROM 13. In the case of a printer without the external memory 14 such as a hard disk or the like, information or the like which is used on a host computer is stored in a data ROM of the ROM 13. The CPU 12 can execute a communicating process with the host computer via an input unit 18 and notify the host computer 3000 of the information or the like in the printer. A RAM 19 is a RAM which functions as a main memory, a work area, or the like of the CPU 12 and is constructed so that a memory capacity can be expanded by an option RAM which is connected to an expansion port (not shown). The RAM 19 is used as an output information rasterizing area, an environment data storing area, an NVRAM, or the like. An access of the external memory 14 such as hard disk (HD), IC card, or the like mentioned above is controlled by a memory controller (MC) 20. The external memory 14 is connected as an option and used for storing font data, an emulation program, form data, and the like and spooling the rasterized print data. Switches for operation, an LED display, and the like are arranged on the operation panel 18. Each module shown in Fig. 19, which will be explained hereinlater, has been stored in the foregoing HD.

As for the external memory 14 mentioned above, the number of memories is not limited to one but the apparatus can be also constructed in a manner such that a plurality of external memories 14 are provided and a plurality of memories such as option card to which built-in fonts have been added, memory for spooling the rasterized print data, and an external memory in which a program for interpreting printer control languages of different language systems have been stored can be connected. Further, the apparatus can be also constructed in a manner such that it has an NVRAM (not shown) and printer mode set information from an operation panel 1501 is stored therein.

Fig. 2 is a constructional diagram of a typical printing process in the host computer to which printing apparatuses such as printers or the like are connected directly or via a network. An application 201, a graphic engine 202, a printer driver 203, and a system spooler 204 are program modules which exist as files stored in the external memory 11. When those program module is executed, it is loaded into the RAM 2 by the OS or a module which uses such a module and executed. The application 201 and the printer driver 203 can be added to the FD of the external memory 11 or a CD-ROM (not shown) or to the HD of the external memory 11 via the network (not shown). Although the application 201 held in the external memory 11 is loaded into the RAM 2 and executed, when the application 201 allows the printer 1500 to execute the print, the output (drawing) is executed by using the graphic engine 202 which has similarly been loaded into the RAM 2 and can be executed.

The graphic engine 202 similarly loads the printer driver 203 prepared every printing apparatus into the RAM 2 from the external memory 11 and sets an output of the application 201 into the printer driver 203. The graphic engine 202 converts a GDI (Graphic Device Interface) function received from the application 201 into a DDI (Device Drive Interface) function and outputs the DDI function to the printer driver 203. On the basis of the DDI function received from the graphic engine 202, the printer driver 203 converts it into a control command which can be recognized by the printer, for example, PDL

(Page Description Language). The converted printer control command is supplied to the system spooler 204 loaded in the RAM 2 and outputted as print data to the printer 1500 via an interface 21 by the OS.

In addition to the print system comprising the printer and the host computer shown in Fig. 2, the print system of the embodiment further has a construction such that the print data supplied from the application is spooled once by intermediate code data as shown in Fig. 3.

Fig. 3 is a diagram obtained by expanding the system of Fig. 2 and has a construction such that when a print command is sent from the graphic engine 202 to the printer driver 203, a spool file 303 comprising intermediate codes is formed once. In the system of Fig. 2, the application 201 is released from the printing process at a point of time when the printer driver 203 has completely finished the conversion from all print commands from the graphic engine 202 into printer control commands. On the other hand, in the system of Fig. 3, it is released at a point of time when the spooler 302 converts all print commands into intermediate code data and outputs it to the spool file 303. Ordinarily, a time which is required in the latter method is shorter than that in the former method. In the system shown in Fig. 3, contents in the spool file 303 can be modified. Thus, it is possible to realize functions which the application does not have, that is, the functions such as enlargement/reduction, N-up print for reducing a plurality of pages and printing them into one page, and the like for the print data sent from the application.

To realize the above objects, the system of Fig. 2 is expanded so as to spool the print command by the intermediate code data as shown in Fig. 3. To modify the print data, ordinarily, the setting is performed from an window that is provided by the printer driver 203 and the printer driver 203 holds the set contents into the RAM 2 or the external memory 11.

Details of Fig. 3 will be described hereinbelow. As shown in Fig. 3, according to the expanded processing system, a dispatcher 301 receives the print commands from the graphic engine 202. If the print command received by the dispatcher 301 from the graphic engine 202 is a print command issued from the application 201 to the graphic engine 202, the dispatcher 301 loads a spooler 302 stored in the external memory 11 into the RAM 2 and sends the print command to the spooler 302 instead of the printer driver 203.

The spooler 302 converts the received print command into an intermediate code and outputs it to the spool file 303. The spooler 302 obtains the modification settings regarding the print data set for the printer driver 203 from the printer driver 203 and holds them into the spool file 303. Although the spool file 303 is formed as a file into the external memory 11, it can be also formed in the RAM 2. The spooler 302 further loads a spool file manager 304 stored in the external memory 11 into the RAM 2 and notifies the spool file manager 304 of a forming situation of the spool file 303. After that, the spool file manager 304 discriminates whether the print can be performed or not in accordance with contents of the modification settings regarding the print data stored in the spool file 303.

If the spool file manager 304 determines that the print can be performed by using the graphic engine 202, it loads a despooler 305 stored in the external memory 11 into the RAM 2 and instructs the despooler 305 to execute the printing process of the intermediate code described in the spool file 303.

The despooler 305 modifies the intermediate code included in the spool file 303 in accordance with the contents of the modification settings included in the spool file 303 and outputs it again via the graphic engine 202.

If the print command received by the dispatcher 301 from the graphic engine 202 is a print command issued from the despooler 305 to the graphic engine

202, the dispatcher 301 sends the print command to the printer driver 203 instead of the spooler 302.

The printer driver 203 forms a printer control command and outputs it to the printer 1500 via the system spooler 204.

Fig. 5 shows a correlation diagram between friendly names of printer drivers and hexadecimal numbers which are formed by CRC.exe. Fig. 6 shows an example of an INF file in Windows (registered trademark) 2000 which is used in the embodiment. Fig. 7 shows an example of an INF file in Windows NT40 (registered trademark) which is used in the embodiment. Fig. 8 shows necessity of system re-activation after the installation. A method of installing printer drivers from a printer folder of the OS which is preferable for the embodiment will be described in detail hereinbelow with reference to Figs. 4 to 8.

Fig. 4 shows a schematic diagram of a printer driver installing method for explaining a whole construction of the embodiment. A method which shows the whole construction of the embodiment and whereby a name of a module set of a printer driver which is installed is changed in accordance with a predetermined rule upon installation will be described with reference to Fig. 4. The driver module set is included in an installation set and denotes a set of driver modules which forms a set in accordance with a predetermined relation.

A rename process of the driver as an element technique which is frequently used in the embodiment will be described first hereinbelow. In the embodiment, the names of the driver modules are changed in accordance with a predetermined rule upon installation. Hereinafter, "change the name" is also referred to as "rename". The renaming process is realized by a method whereby when a driver is installed into the host computer 3000 in Fig. 1, the installer stored in the HDD 11 in the host computer 3000 is loaded into the RAM 2 and by controlling the API (not shown) of the OS which is similarly loaded into the RAM 2, control is made so as to rename the name of the driver module. Naturally, in the case of a system such that the installer as an application directly permits the rename, the name can be directly renamed.

Subsequently, an example of the renaming process will be shown. For example, assuming that AAA.dll, BBB.dll, and CCC.dll as a module set of the driver are targets of the rename upon installation of every model type, for example, when a printer driver of a model type "111" is installed, they are copied by names AAA_111.dll, BBB_111.dll, and CCC_111.dll into the system directory of Windows (registered trademark), respectively. When a printer driver of a model type "222" is installed, they are copied by names AAA_222.dll, BBB_222.dll, and CCC_222.dll into the system directory of the OS, respectively. When a printer driver of a model type "333" is installed, they are copied by names AAA_333.dll, BBB_333.dll, and CCC_333.dll into the system directory of the OS, respectively.

In this instance, assuming that DDD.dll and EEE.dll are modules as rename non-targets upon installation of every model type, when the printer driver of the model type "111" is installed and when the printer driver of the model type "222" is installed, they are copied as DDD.dll and EEE.dll into the system directory of the OS, respectively.

Definitions of the printer driver module as a rename target and the printer driver module as a rename non-target will be described here. The printer driver module as a rename target is a module which provides fundamental functions of the driver and is a module which is continuously loaded into the system of the OS once the print is executed. For example, modules such as graphics driver (CNP5EE.DLL corresponds to it in the embodiment), user interface driver (CNP5EEUI.DLL corresponds to it in the embodiment), and resource file (CNP5E809.DLL corresponds to it in the embodiment) correspond to those modules. The printer driver module as a rename non-target is a model type common file such as color profile or help file and is a module which is unloaded from the system of the OS after completion of the print.

Fig. 5 is a table showing by which names the printer driver modules as rename targets are copied into the system directory of the OS. Specifically speaking, a hexadecimal number of four digits written in this table is added after each module name incorporated in the driver set. In this instance, "F0E5" is added to the module of a driver name "PrinterMakerA iR1600-2000 PCL5e", "617E" is added to the module of a driver name "PrinterMakerA iR2200-3300 PCL5e", "CA5C" is added to the module of a driver name "PrinterMakerA iR400 PCL5e", "9926" is added to the module of a driver name "PrinterMakerA iR5000-6000 PCL5e", "1579" is added to the module of a driver name "PrinterMakerA iR5000-6000-L1 PCL5e", "FAEA" is added to the module of a driver name "PrinterMakerA iR7200 PCL5e", and "D6F1" is added to the module of a driver name "PrinterMakerA iR7200-M1 PCL5e", respectively. "PrinterMakerA" denotes a name of a printer maker. "iR1600-2000" denotes a model name of a printer. "PCL" denotes a printer language. It is assumed that the driver names have been arranged in this order.

For example, in the case of the "PrinterMakerA iR1600-2000 PCL5e" driver, the graphics driver is "CNP5EE_F0E5. DLL", the user interface driver is "CNP5EEUI_F0E5. DLL", and the resource file is "CNP5E809_F0E5. DLL", respectively. In the case of the "PrinterMakerA iR7200 PCL5e" driver, the graphics driver is "CNP5EE_FAEA. DLL", the user interface driver is "CNP5EEUI_FAEA. DLL", and the resource file is "CNP5E809_FAEA. DLL", respectively.

In the case of the "PrinterMakerA iR5000-6000-L1 PCL5e" driver, those drivers are determined as follows: the graphics driver is "CNP5EE_1579. DLL"; the user interface driver is "CNP5EEUI_1579. DLL"; and the resource file is "CNP5E809_1579. DLL", respectively.

Numerals of those lower four digits (which can be expressed by four bits in the binary notation) are formed by a program tool called "CRC. exe". By using this tool, hexadecimal numbers of arbitrary four digits can be formed from the friendly name of the printer driver. For example, the reason why the end of "PrinterMakerA iR1600-2000 PCL5e" is not determined to be "111" from the beginning and the reason why the end of "PrinterMakerA iR7200 PCL5e" is not determined to be "222" from the beginning are because if they are determined from the beginning, hard coding such as a rename routine or the like having a table of the model names and the module change names is needed in the program and, each time a new model type is added, the program has to be corrected (added). It is also because as compared with the above case, if the driver module name is formed from the friendly name of the printer driver by using the tool, there is an effect such that a new model type can be added merely by correcting an INF file as an external file, which will be explained hereinlater.

Fig. 19 is a diagram showing an example of the driver installer which has been stored in the HDD 11 of the host computer 3000 in Fig. 1, is read out into the RAM, and is executed and an example of the modules of the OS. Reference numeral 1901 denotes a driver module set in which drivers of a plurality of model types and names have been stored. The driver installer is a control program for installing the driver, reads out an INF file 1907 as a setting file for controlling the driver installer, and controls a system installer 1905 and a renaming unit 1906 of the OS in accordance with the read-out INF file 1907. Although "renaming unit" has been written here, a handling function of a general file of the OS, for example, functions such as file creation, file name change, file movement, and the like in the file system of the OS are referred to as a renaming unit

1906. By calling an API of the system installer 1905, a driver installer 1902 controls the system installer 1905, copies the driver module set into a system directory 1904 which is managed by the OS, and registers information of the module which is installed into a registry 1903. The registry is a data structure which is managed by the OS. By reading out such an area, the OS recognizes which driver is installed and how it has been set.

Subsequently, a describing method of the INF file for changing the name of the module set of the printer driver upon installation will be described. In this instance, when a predetermined character string "CRC. exe" is inputted, a hexadecimal number corresponding to the model name is formed by using a tool for forming the hexadecimal numbers which correspond to the predetermined character string in a one-to-one correspondence relational manner. It is now assumed that the friendly name is expressed by the printer maker name, model name, and printer language (printer language version name). The INF files in Figs. 6 and 7 have been described for a specific OS. The INF files have been stored in the HDD 11 in Fig. 1 and correspond to 1907 in Fig. 19. The INF file is read out to the driver installer in response to the execution of the driver installer 1902. Samples of the INF files are as shown in Figs.

6 and 7. When necessary portions among them are extracted, they are as follows.

```
;Identification #PCL5eUK
;IR8500: E287
;LBP-2000: 441B
[IR8500]
CNP5EE_E287. DLL CNP5EE. DLL
CNP5EEUI_E287. DLL CNP5EEUI. DLL
CNP5E809_E287. DLL CNP5E809. DLL
[IR8500_DATA]
DriverFile = CNP5EE_E287. DLL
ConfigFile = CNP5EEUI_E287. DLL
[LB2000]
CNP5EE_441B. DLL CNP5EE. DLL
CNP5EEUI_441B. DLL CNP5EEUI. DLL
CNP5E809_441B. DLL CNP5E809. DLL
[LB2000_DATA]
DriverFile = CNP5EE_441B. DLL
ConfigFile = CNP5EUI_441B. DLL
```

Explanation is added with respect to the foregoing INF file. IR8500 (iR8500) and LB2000 denote model names of certain printers. A state where the number E287 is made to correspond to IR8500 and the number 441B is made to correspond to LBP-2000 is shown here. Those numbers are formed by using

"CRC.exe".

A character string surrounded by next square brackets [ ] is called a label. Contents of the label [IR8500] are confirmed in order to explain the model type IR8500. When the first line of the set contents of [IR8500] is read, the module written on the right side, that is, "CNP5EE. DLL" denotes a name of an original driver module incorporated in the driver set and the name written on the left side, that is, "CNP5EEUI_E287. DLL" denotes a name which is obtained after the renaming and copied to the system directory of the OS. That is, it means that the module "CNP5EE. DLL" stored in the installation set at present is copied as a name "CNP5EEUI_E287. DLL" and installed into the directory in which the module is installed. This is true of the modules (CNP5EEUI. DLL, CNP5E409. DLL) described on the second and third lines of the set contents of [IR8500].

Subsequently, set contents of the label [LB2000] will be examined in order to consider the model type LB2000. "CNP5EE. DLL" is disclosed so as to be copied as "CNP5EE_441B. DLL". This is true of the modules (CNP5EEUI. DLL, CNP5E409. DLL) disclosed on the second and third lines of the set contents of [LB2000].

By describing as mentioned above by an amount of only the corresponding model types, with respect to one module in the module set included in the driver module set, even if it is the common module, the module which is copied to the system directory of the OS is renamed and copied the number of times as many as the number of model types (CNP5EE_441B. DLL). That is, when explaining with respect to "CNP5EE. DLL", it is renamed to the module name "CNP5EEUI_E287. DLL" in the case of IR8500 and to the module name "CNP5EE_441B. DLL" in the case of LB2000 and, thereafter, it is copied to the system directory of the OS. Even in the case of the modules which have conventionally been managed as a common module in a plurality of model types, they are renamed to the different module names every model type and installed. Therefore, even if the version of the driver of the specific model type is upgraded, the driver of another model type which is not concerned with the version-up is not overwritten or the like and is not subjected to influence which the user does not desire.

Fig. 8 is a diagram showing that since a name of a rename target module is changed upon installation, the module set constructing the printer driver is installed every model type and the module sets are not mutually interfered, so that the necessity of the re-activation of the system after the installation decreases. It is one of the objects of the embodiment and one of the disadvantages of the format in which a plurality of corresponding model types are incorporated in one installation set is solved. If a plurality of corresponding model types are incorporated in one installation set, the existing module which has already been installed in the PC is also overwritten although the new printer driver is installed, so that it is necessary to re-activate the system after the installation. However, the following advantages are obtained by providing the invention of the embodiment.

As shown in a table of Fig. 8, if a "PrinterMakerA iR7200 PCL5ev5.30" driver as the same version is installed into a personal computer in which a "PrinterMakerA iR8500 PCL5ev5.30" driver has been installed, the module which is a fundamental function of the driver and is continuously loaded into the system of the OS once the print is executed in the case of the "PrinterMakerA iR8500 PCL5ev5.30" driver and that of the "PrinterMakerA iR7200 PCL5ev5.30" driver are made different by applying the embodiment. Therefore, the re-activation after the installation of the "PrinterMakerA iR7200 PCL5ev5.30" driver is unnecessary.

The module which is a fundamental function of the driver and is continuously loaded into the system of the OS once the print is executed even if a "PrinterMakerA iR7200 PCL5ev5.40" driver as a different version is installed in the same environment as that mentioned above in the case of the "PrinterMakerA iR8500 PCL5ev5.30" driver and that of the "PrinterMakerA iR7200 PCL5ev5.40" driver are made different by applying the embodiment of the invention. Therefore, the re-activation after the installation of the "PrinterMakerA iR7200 PCL5ev5.40" driver is unnecessary either.

The name of the module set of the printer driver to be installed is changed in accordance with the predetermined rule upon installation as mentioned above, so that not only the driver per model type of high developing efficiency can be formed but also the undesirable system re-activation can be minimized.

Modifications of the first embodiment will be described hereinbelow. In each of the following embodiments, a prerequisite of the installation is similar to that mentioned in the first embodiment, particularly, with respect to Figs. 1 to 4 and Fig. 19. Therefore, portions different from the first embodiment will be mainly explained.

(Second embodiment)

Since the prerequisite of the installation system is as mentioned in the first embodiment of Figs. 1 to 4 and Fig. 19, its disclosure is omitted hereinbelow. In the following embodiment, another example of rename is disclosed. Fig. 9 is a table showing by which names the rename target printer driver modules are copied into the system directory of the OS. In the table, only graphics drivers are shown with respect to the driver target files. In Fig. 9, a rename format is "post-rename name = pre-rename name + "_" + model name".

For example, in the case of the "PrinterMakerA iR1600-2000 PCL5e" driver, the rename format is as follows. First, the graphics driver is "CNP5EE_PrinterMakerA iR1600-2000 PCL5e. DLL", the user interface driver is "CNP5EEUI_PrinterMakerA iR1600-2000 PCL5e. DLL", and the resource file is "CNP5E809_PrinterMakerA iR1600-2000 PCL5e. DLL", respectively.

In the case of the "PrinterMakerA iR7200 PCL5e" driver, the rename format is as follows. First, the graphics driver is "CNP5EE_PrinterMakerA iR7200 PCL5e. DLL", the user interface driver is "CNP5EEUI_PrinterMakerA iR7200 PCL5e. DLL", and the resource file is "CNP5E809_PrinterMakerA iR7200 PCL5e. DLL", respectively.

In the case of the "PrinterMakerA iR5000-6000-L1 PCL5e" driver, the graphics driver is "CNP5EE_PrinterMakerA iR5000-6000-L1 PCL5e. DLL", the user interface driver is "CNP5EEUI_PrinterMakerA iR5000-6000-L1 PCL5e. DLL", and the resource file is "CNP5E809_PrinterMakerA iR5000-6000-L1 PCL5e. DLL", respectively.

The position where the model name is added, a character string for connecting the pre-rename name and the model name, and the like are not important. It is important to add the model name. For example, it is also possible to use a method such as "post-rename name = pre-rename name + "@" + model name" or a method such as "post-rename name = model name + "_" + pre-rename name".

The reason why the name after the rename (post-rename name) is not determined from the beginning is because if it has been predetermined, hard coding is necessary in the program and the program has to be corrected (added) each time the model type is added. As compared with the above case, if the driver module name is formed from the model name of the printer driver, a new model type can be added merely by correcting the INF file as an external file and there is no need to correct the program.

Subsequently, a describing method of the INF file for changing the name of the module set of the printer driver upon installation will be described. Samples of the INF files are as shown in Figs. 10 and 11. When necessary portions among them are extracted, they are as follows.

```
[PrinterMakerA]
"PrinterMakerA iR8500 PCL5e" = IR8500, PrinterMakerA
           IR8500 059D
[IR8500]
CopyFiles = IR8500_FILESPCL5E_FILES
DataFile = IR8500PK. XPD
DataSection = IR8500_DATA
[IR8500_DATA]
DriverFile = "CNP5EE_PrinterMakerA iR8500 PCL5e. DLL"
ConfigFile = "CNP5EEUI_PrinterMakerA iR8500 PCL5e.
             DLL"
HelpFile = CNP5EE. HLP
[IR8500_FILES]
"CNP5EE_PrinterMakerA iR8500 PCL5e. DLL" CNP5EE. DLL
"CNP5EEUI_PrinterMakerA iR8500 PCL5e. DLL" CNP5EEUI.
           DLL
"CNP5E809_PrinterMakerA iR8500 PCL5e. DLL" CNP5E809.
           DLL
```

In [IR8500_FILES] mentioned above, the name written on the right side indicates the name of the original driver module incorporated in the driver set and the name written on the left side indicates the post-rename name which is copied into the system directory of Windows (registered trademark). By describing the names of the number as many as the number of corresponding model types as mentioned above, even if there is one kind of module set incorporated in the driver set, as modules which are copied into the system directory of the OS, the modules of the number as many as the number of model types exist.

(Third embodiment)

Fig. 12 is a table showing by which names the printer driver modules as rename targets are copied into the system directory of Windows (registered trademark). By adding GUID (Global Unique ID) to each rename target module, its name is changed. Since the GUID is formed every installation, it is a unique ID every installation. Therefore, the rename target module is not always renamed in accordance with the file name after the rename shown in Fig. 12 each time. Fig. 12 shows an example of the rename format. In Fig. 12, the rename format is

"post-rename name = pre-rename name + "_" + GUID"

For example, the GUID as a preferred example of information having uniqueness can be formed by combining an address obtained by encoding an MAC address of a network card of the host computer and execution start time of the installation or the like.

For instance, in the case of the "PrinterMakerA iR1600-2000 PCL5e" driver, the graphics driver is "CNP5EE_1B3ADB36-3C65-4f8d-AFC9-AFB020463D5D. DLL", the user interface driver is "CNP5EEUI_1B3ADB36-3C65-

4f8d-AFC9-AFB020463D5D. DLL", and the resource file is "CNP5E809_1B3ADB36-3C65-4f8d-AFC9-AFB020463D5D. DLL", respectively.

In the case of the "PrinterMakerA iR7200 PCL5e" driver, the graphics driver is "CNP5EE_D06A99AC-4BB7-44ed-AEC3-BEF2DBCB5BBC. DLL", the user interface driver is "CNP5EEUI_D06A99AC-4BB7-44ed-AEC3-BEF2DBCB5BBC. DLL", and the resource file is "CNP5E809_D06A99AC-4BB7-44ed-AEC3-BEF2DBCB5BBC. DLL", respectively.

In the case of the "PrinterMakerA iR5000-6000-L1 PCL5e" driver, the graphics driver is "CNP5EE_590C71FD-D88A-4e90-B72A-C40CBB73D28D. DLL", the user interface driver is "CNP5EEUI_590C71FD-D88A-4e90-B72A-C40CBB73D28D. DLL", and the resource file is "CNP5E809_590C71FD-D88A-4e90-B72A-C40CBB73D28D. DLL", respectively.

It is one of the important points that the model name and the version number are added rather a character string or the like connecting the pre-rename name and the GUID. For example, it is also possible to use a method such as "post-rename name = pre-rename name + "@" + GUID" or a method such as "post-rename name = GUID + "_" + pre-rename name".

Since the GUID is formed every installation, it is guaranteed that the post-rename name is unique. When the installer forms the GUID and renames and installs each module in accordance with the foregoing rule, it writes the GUID used for the rename into the registry. Since the post-rename name of the printer driver is dynamically and uniquely determined, it cannot be predetermined from the beginning. Therefore, the name of each of the renamed modules can be obtained by referring to the GUID written in the registry by the installer upon installation. Consequently, the correction of the program for adding the new model type is unnecessary.

(Fourth embodiment)

Fig. 13 is a table showing by which names the rename target printer driver modules are copied into the system directory of the OS. In the table, only graphics drivers of version 5.30 are shown with respect to the driver target files.

In Fig. 13, a rename format is "post-rename name = pre-rename name + "_" + model name + "_" + version number".

For example, in the case of the "PrinterMakerA iR1600-2000 PCL5e" driver, the graphics driver (version 5.30) is "CNP5EE_PrinterMakerA iR1600-2000 PCL5e_530. DLL", the user interface driver (version 5.30) is "CNP5EEUI_PrinterMakerA iR1600-2000 PCL5e_530. DLL", and the resource file (version 5.30) is "CNP5E809_PrinterMakerA iR1600-2000 PCL5e_530. DLL", respectively.

In the case of the "PrinterMakerA iR7200 PCL5e" driver, the graphics driver is "CNP5EE_PrinterMakerA iR7200 PCL5e_530. DLL", the user interface driver is "CNP5EEUI_PrinterMakerA iR7200 PCL5e_530. DLL", and the resource file is "CNP5E809_PrinterMakerA iR7200 PCL5e_530. DLL", respectively.

In the case of the "PrinterMakerA iR5000-6000-L1 PCL5e" driver, the graphics driver (version 5.40) is "CNP5EE_PrinterMakerA iR5000-6000-L1 PCL5e_540. DLL", the user interface driver (version 5.40) is "CNP5EEUI_PrinterMakerA iR5000-6000-L1 PCL5e_540. DLL", and the resource file (version 5.40) is "CNP5E809_PrinterMakerA iR5000-6000-L1 PCL5e_540. DLL", respectively.

The position where the model name is added, the position where the version number is added, a character string for connecting the pre-rename name and the model name, and the like are not important. It is one of the important points that the model name and the version number are added. For example, it is also possible to use a method such as "post-rename name = pre-rename name + "@" + model name + "@" + version number" or a method such as "post-rename name = model name + "_" + version number + "_" + pre-rename name".

The reason why the name after the rename (post-rename name) is not determined from the beginning without using the model name and the version number is because if it has been predetermined, hard coding is necessary in the program and the program has to be corrected (added) each time the model type is added. As compared with the above case, if the driver module name is formed from the model name and the version number of the printer driver, a new model type can be added merely by correcting the INF file as an external file and there is no need to correct the program.

Subsequently, a describing method of the INF file for changing the name of the module set of the printer driver upon installation will be described. Samples of the INF files are as shown in Figs. 14 and 15. When necessary portions among them are extracted, they are as follows.

```
[PrinterMakerA]
"PrinterMakerA iR8500 PCL5e" = IR8500, PrinterMakerA
            IR8500 059D
[IR8500]
CopyFiles = IR8500_FILESPCL5E_FILES
DataFile = IR8500PU. XPD
DataSection = IR8500_DATA
[IR8500_DATA]
DriverFile = "CNP5EE_PrinterMakerA iR8500 PCL5e_530.
              DLL"
```

```
ConfigFile = "CNP5EEUI_PrinterMakerA iR8500 PCL5e_530.
             DLL"
HelpFile = CNL4J. HLP
[IR8500_FILES]
"CNP5EE_PrinterMakerA iR8500 PCL5e_530. DLL" CNP5EE.
           DLL
"CNP5EEUI_PrinterMakerA iR8500 PCL5e_530. DLL"
           CNP5EEUI. DLL
"CNP5E409_PrinterMakerA iR8500 PCL5e_530. DLL"
           CNP5E409. DLL
```

In [IR8500_FILES] mentioned above, the name written on the right side indicates the name of the original driver module incorporated in the driver set and the name written on the left side indicates the post-rename name which is copied into the system directory of the OS. By describing the names of the number as many as the number of corresponding model types as mentioned above, even if there is one kind of module set incorporated in the driver set, as modules which are copied into the system directory of the OS, the modules of the number as many as the number of model types exist.

Fig. 8 is a diagram showing a state where when the names of the rename target modules are changed upon installation, the module set constructing the printer driver is installed every model type and they are not mutually interfered, so that the necessity of the system re-activation after the installation is reduced. It shows that one of the disadvantages of the format in which a plurality of corresponding model types are incorporated in one installation set can be solved. If a plurality of corresponding model types are incorporated in one installation set, the existing module which has already been installed in the PC is also overwritten although the new printer driver is installed, so that it is necessary to re-activate the system after the installation. However, such a problem is solved by the invention.

As shown in the table of Fig. 8, if the "PrinterMakerA iR7200 PCL5ev530" driver as the same version is installed into a personal computer in which the "PrinterMakerA iR8500 PCL5ev5.30" driver has been installed, the module which is a fundamental function of the driver and is continuously loaded into the system of Windows (registered trademark) once the print is executed in the case of the "PrinterMakerA iR8500 PCL5ev5.30" driver and that of the "PrinterMakerA iR7200 PCL5ev5.30" driver are different. Therefore, the re-activation after the installation of the "PrinterMakerA iR7200 PCL5ev5.30" driver is unnecessary. The module which is a fundamental function of the driver and is continuously loaded into the system of Windows (registered trademark) once the print is executed even if the "PrinterMakerA iR7200 PCL5ev5.40" driver as a different version is installed in the same environment as that mentioned above in the case of the "PrinterMakerA iR8500 PCL5ev5.30" driver and that of the "PrinterMakerA iR7200 PCL5ev5.40" driver are made different. Therefore, the re-activation after the installation of the "PrinterMakerA iR7200 PCL5ev5.40" driver is unnecessary either.

The name of the module set of the printer driver to be installed is changed in accordance with the model name and the version number of the printer driver as mentioned above, so that not only the driver per model type of high developing efficiency can be formed but also the undesirable system re-activation can be minimized.

As described above, according to the installing apparatus of the printer driver of the fourth embodiment, when the printer driver is installed into the information processing apparatus such as a personal computer or the like, the name of the module set to be installed is changed in accordance with the model name and the version number of the driver upon installation. Thus, the driver module set which is common to all corresponding model types can be installed by a different name per model type and per version. Therefore, not only the driver per model type and per version of very high developing efficiency can be formed but also the undesirable system re-activation can be almost completely prevented.

(Fifth embodiment)

Fig. 16 is a table showing by which names the rename target printer driver modules are copied into the system directory of Windows (registered trademark). In Fig. 16, a rename format is "post-rename name = pre-rename name + "_" + numerical value described in printer driver data file". The printer driver data file is a file in which information to set the printer driver has been stored. A table corresponding to each driver module name for renaming mentioned above has been stored here.

For example, in the case of the "PrinterMakerA iR1600-2000 PCL5e" driver, the graphics driver is "CNP5EE_0. DLL", the user interface driver is "CNP5EEUI_0. DLL", and the resource file is "CNP5E809_0. DLL", respectively.

In the case of the "PrinterMakerA iR7200 PCL5e" driver, the graphics driver is renamed to "CNP5EE_5. DLL", the user interface driver is renamed to "CNP5EEUI_5. DLL", and the resource file is renamed to "CNP5E809_5. DLL", respectively.

In the case of the "PrinterMakerA iR5000-6000-L1 PCL5e" driver, the graphics driver is "CNP5EE_4.

DLL", the user interface driver is "CNP5EEUI_4. DLL", and the resource file is "CNP5E809_4. DLL", respectively.

Those names obtained after the name change are also described in the driver data file. The name obtained after the name change which is described in the printer driver data file and the name obtained after the name change which is described in the INF file have to be certainly coincident. Consequently, since the printer driver can read out the names obtained after the name change from the driver data file, there is no need to hard-code the names obtained after the name change into the program. In other words, there is no need to correct (add) the program each time the model type is added. A new model type can be added merely by correcting the INF file as an external file and the driver data file.

Subsequently, a describing method of the INF file for changing the name of the module set of the printer driver upon installation will be described. Samples of the INF files are as shown in Figs. 17 and 18. When necessary portions among them are extracted, they are as follows.

```
[IR8500]
IR8500PK. XPD
CNP5EE_8. DLL CNP5EE. DLL
```

CNP5EEUI_8. DLL CNP5EEUI. DLL
CNP5E809_8. DLL CNP5E809. DLL
[LB2000]
LB2000PK. XPD
CNP5EE_15. DLL CNP5EE. DLL
CNP5EEUI_15. DLL CNP5EEUI. DLL
CNP5E809_15. DLL CNP5E809. DLL

The name written on the right side indicates the name of the original driver module incorporated in the driver set and the name written on the left side indicates the post-rename name which is copied into the system directory of the OS. The printer driver data file mentioned above is "IR8500PK. XPD" or "LB2000PK. XPD". By describing the names of the number as many as the number of corresponding model types as mentioned above, even if there is one kind of module set incorporated in the driver set, as modules which are copied into the system directory of the OS, the modules of the number as many as the number of model types exist.

(Other embodiments)

Although the case of installing the printer driver from a printer folder of the OS has been mentioned in the foregoing embodiments, it can be also installed by an installer which uniquely provides it. Specifically speaking, if the driver module name is changed on the basis of the driver INF in the installer which uniquely provides it and the driver module whose name has been changed is installed by the API of the OS, the installation per model type mentioned in the embodiments can be performed and substantially the same function as the installation from the printer folder of the OS can be provided.

Each of the processes in the embodiments can be executed by each apparatus such as client device, printer, or the like in accordance with a program which is installed from the outside. In such a case, the invention is also applied to a case where an information group including the program is supplied to the host computer from a memory medium such as CD-ROM, flash memory, FD, or the like or from an external memory medium via a network.

As mentioned above, naturally, the objects of the invention are accomplished by a method whereby the memory medium in which program codes of software for realizing the functions of the embodiments mentioned above have been recorded is supplied to a system or an apparatus or they are downloaded from an external server (not shown), so that a computer (or a CPU or an MPU) of the system or the apparatus reads out the program codes stored in the memory medium and executes them.

In such a case, the program codes themselves read out from the memory medium realize the novel functions of the invention and the memory medium in which the program codes have been stored constructs the invention. As a memory medium for supplying the program codes, for example, a floppy disk, a hard disk, an optical disk, a magnetooptic disk, a DVD, a CD-ROM, a magnetic tape, a non-volatile memory card, a ROM, an EEPROM, or the like can be used.

Naturally, the invention incorporates not only a case where a computer executes the read-out program codes, so that the functions of the embodiments mentioned above are realized but also a case where an OS (Operating System) or the like which is operating on a computer executes a part or all of actual processes on the basis of instructions of the program codes and the functions of the embodiments mentioned above are realized by those processes. Further, naturally, the invention incorporates a case where the program codes read out from the memory medium are written into a memory provided for a function expanding board inserted in a computer or a function expanding unit connected to a computer and, thereafter, a CPU or the like provided for the function expanding board or the function expanding unit executes a part or all of actual processes on the basis of instructions of the program codes and the functions of the embodiments mentioned above are realized by those processes.

Naturally, as shown in Fig. 8, the necessity of the re-activation is reduced by using the first to fifth embodiments mentioned above. For example, according to the installing apparatus of the printer driver in each of the embodiments of the invention, when the printer driver is installed into the information processing apparatus such by a personal computer or the like, the name of the module set to be installed is changed in accordance with the predetermined rule upon installation. Thus, the driver module set which is common to all of the corresponding model types can be installed as a different name per model type. Therefore, not only the driver per model type of high developing efficiency can be formed but also the undesirable system re-activation can be minimized.

As described above, according to one aspect of the invention, since the identification information of the driver module set is changed by using the identification information having uniqueness which is formed upon installation, the driver's undesirable overwrite updating process can be prevented.